IN THE SPECIFICATION

Please amend the specification as follows:

[0002] Graphics applications, and particularly three dimensionally (3D) graphics applications have long been one of the most processing intensive activities performed by personal computers. To improve graphics processing capabilities, graphics co-processors have proliferated and are widely available on most modern day personal computers. Graphics coprocessors are specialized integrated circuits designed to quickly perform processing intensive tasks required by graphic applications.

[0004] It is frequently the case when a texture image is applied to an object in a final rendering, there is disparity between a number of sample texture elements (texels) and the source texture image and the number of picture elements (pixels) to which the image is mapped. When the number of texels in a given range is less than the number of pixels, then the texture is required to be upsampled. When upsampling a texture, a scheme must be used to fill intermediate values. This scheme is referred to herein as "texture filtering" and has largely been performed by a fixed function state machine.

[0004.1] Most current graphics coprocessor support four types of texture filtering; point sampling, bilinear filtering, trilinear filtering and anisotropic filtering. As the filtering methods become increasingly complex, the state machine required to perform them becomes increasingly complex and requires increased real estate within the graphics coprocessor. This coupled with the fact that uses for texture data continues to expand, for example, texture data is being used for lighting and other surface properties in addition to color, renders the commonly employed linear interpolation inefficient or even insufficient.

[0006] Figure 1~~A~~ is a block diagram of a system of one embodiment of the invention.

~~[0007] Figure 1B is a diagram of texture sampling in one embodiment of the invention.~~

[00010] Figure 1~~A~~ is a block diagram of a system of one embodiment of the invention. A host processor 100 is coupled by a bus 102 to a memory 104. A graphics coprocessor 106 is also coupled to the bus 102. Additionally, graphics coprocessor 106 may be coupled to memory 104 by an accelerated graphics port (AGP) 112. The AGP may adhere to Accelerated Graphics Port AGP V3.0 Interface Specification Rev. 1.0 published September 2002 (hereinafter the AGP Specification). AGP 112 allows rapid access to graphics data residing in memory 104. Also coupled to the bus are framebuffer 108 and display 110. In some embodiments, framebuffer 108 may be contained within memory 104. Graphics coprocessor 106 includes pixel processing pipeline 120. Within the pixel processing pipeline 120 is a vertex processing module 122, primitive assembly module 124, a fragment processing module 126 and a framebuffer processing module 128. Vertex processing module 122 in operation receives vertex data, which may include, for example, 3D positional information, color information and other similar information related to vertices in the graphic image. In one embodiment, vertex data is of the form V = X, Y, Z, T_u, T_v, RGB. In this expression, X, Y, Z are the three dimensional cartesian coordinates of the vertex, T_u and T_v are the two dimensional coordinate of the corresponding texel in the texture map and RGB are the red, green and blue color values at the vertex. Other forms and contents of vertex data are also contemplated.

[0017] ~~Figure 1B is a diagram of texture sampling in one embodiment of the invention.~~ In one embodiment, sixteen source registers are provided. ~~With~~ In one embodiment, each register corresponding to one texel in a 4 X 4 grid surrounding ~~the~~ a T_uT_v sampling location of ~~the~~ a texture sample point and would correspond to pixels addressed in such a fashion. While T_uT_v ~~mapps~~ may map to a location ~~between texels 5 and 6 and texels 9 and 10~~within the 4 X 4 grid, the contribution of the sixteen texels ~~in the patch~~ to the texture value assigned to T_uT_v may be defined by the texture filtering program. In some embodiments, ~~only~~ a limited number of texels ~~5, 6, 9 and 10~~ provide a contribution. In other embodiments, all sixteen texels may contribute. In still other embodiments, all diagonal ~~prixels~~ pixels in the group may contribute. ~~As illustrated, the programmable nature of the texture filtering module permits robust and flexible texture filtering options.~~

[0019] The actual filtering may be performed by the texture filtering module 130 by loading a desired filtering program into a textured processing core. The filtering program corresponds to a fragment to be processed. Within a region of an image, it may be desirable to apply various effects to the texture data accordingly. Thus, for a particular graphic image, there may be numerous filtering programs employed.

[0019.1] For example, the filter program applied to a shiny part of a leather jacket on an image would likely to be different than the filter program applied to a scuffed part of a leather jacket. By using different programs in e texture filter module, the different effect can be accommodated. The usage of several filter programs during the course of rendering a given scene image is analogous to how, under the current-day fixed-function schemes, the rendering of a given scene may involve switching between the different fixed-function filtering states for different parts of the scene.

[0020] The program employed will influence which of the ~~e.g. 16~~ texels (e.g. 16 for a 4 x 4 grid) are actually sampled to perform the texture filtering . In one embodiment, texture data may be arranged in memory to optimize access to the texels likely to be sampled. For example, assuming the 4 x 4 is grid numbered left to right and top to bottom from 0-15, if the sampling register indicates every fourth texel value is active, the texture data may be stored so that points 1, 5, 9 and 13 are contiguous in memory, and points 2, 6, 10, etc. are contiguous. As another example, where every second texel is active, 1, 3, 5, 7, etc. are contiguous and 2, 4, 8, etc. are contiguous. This arrangement in memory may be performed by the host processor 100 or the graphic coprocessor 106.